COS30018 - Option B - Task 5: Machine Learning 2

The codebase for the model has moved to version v0.4. Currently the program only uses one feature, that being the closing stock price of a company, to predict the closing price for a singular day in the future. The goal of this task is to solve more advanced prediction problems, which include multivariate prediction and multistep prediction.

The multivariate prediction involves using multiple variables. In this instance, that would mean the model using different features (e.g. highest price, opening price, lowest price) to predict the target variable. The multistep prediction involves predicting multiple future values in the time series, not just the following one. In this context, the time series is the daily closing prices collected over the provided date-time period, and the time step is each individual result collected in the time series. Multi-step prediction is important as it will allow us to understand how trends could continue in the future of the data.

A new enhancement was made to the prepare_data function to support both multistep predictions and multivariate input. The function now handles predicting multiple future days by incorporating a steps_ahead parameter, which determines how many future days (k days) to predict. The function loops through the historical data, creating sequences (x_data) of prediction_days length, and collects the corresponding future closing prices (y_data) for steps_ahead days. This outputs x_data and y_data for both single and multiday predictions.

The feature_columns list was updated to include additional features such as Volume, making the model capable of handling multivariate data. This allows the model to learn from more than just the 'Close' price, incorporating other relevant stock features into the prediction.

The build_model function was modified to include steps_ahead as a parameter. The Dense layer's units parameter is now set to steps_ahead, enabling the model to output predictions for multiple future days at once. This improvement allows the model to predict sequences of future closing prices, supporting both single-step and multistep forecasts.

In the previous version (v0.4), the model was limited to making single-step predictions, meaning it could only predict the stock price for the next day. The new implementation introduces the ability to predict multiple future days at once by using the steps_ahead parameter in the prepare_data function and modifying the model's output layer to handle multistep predictions.

```python
def prepare_data(data, feature_columns, prediction_days, steps_ahead=1, multistep=False):
    """
    Prepares data for training an LSTM model by creating sequences.
    - If multistep is True, creates data for multistep prediction.
    """

    # Initializes lists to store input sequences (x_data) and target values (y_data)
    x_data = []
    y_data = []

    if multistep:
        # Creates sequences of historical data for the input and multiple future steps for the output
        for i in range(prediction_days, len(data) - steps_ahead + 1):
            # Takes 'prediction_days' worth of data as input
            x_data.append(data[feature_columns].iloc[i - prediction_days:i].values)
            # Predicts multiple future 'Close' values (steps ahead)
            y_data.append(data['Close'].iloc[i:i + steps_ahead].values)
    else:
        # For just one day ahead (single step)
        for i in range(prediction_days, len(data)):
            # Takes 'prediction_days' worth of data as input
            x_data.append(data[feature_columns].iloc[i - prediction_days:i].values)
            # Predicts the next day's 'Close' value
            y_data.append(data['Close'].iloc[i])

    # Converting to numpy arrays for training
    x_data, y_data = np.array(x_data), np.array(y_data)
    return x_data, y_data
```

Figure 1 - Adjusted Prepare_data function

```python
2 usages
def prepare_data(data, feature_columns, prediction_days):
    """
    Prepares the data for LSTM training.
    """

    x_data = []
    y_data = []

    # Creating sequences for the LSTM model
    # Adds the previous 'prediction_days' days of feature data to x_data
    for i in range(prediction_days, len(data)):
        x_data.append(data[feature_columns].iloc[i - prediction_days:i].values)
        # Adds the current day's closing price to y_data
        y_data.append(data['Close'].iloc[i])

    # Converts the lists to numpy arrays for training
    x_data, y_data = np.array(x_data), np.array(y_data)
    return x_data, y_data
```

Figure 2 - Previous prepare_data function

V0.4 focused on univariate data, which meant using only the 'Close' price for predictions. The new version introduces multivariate support, allowing the model to use multiple features (e.g., Open, High, Low, Volume) for prediction, which can lead to more accurate models with better insight.

```
# Feature columns used in the analysis
feature_columns = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']
```

Figure 3 – Addition of Volume to feature list

The build_model function was updated to handle multistep predictions by modifying the output layer of the neural network. In v0.4 it was always a single value (the next day's closing price), but now it can output multiple values corresponding to the number of future days to predict.

```
# Add final dense layer to output prediction
# For multistep predictions, output size equals steps_ahead
model.add(Dense(units=steps_ahead, activation='linear'))

# Compile the model using the specified optimizer and loss function
model.compile(optimizer=optimizer, loss=loss, metrics=['mean_absolute_error'])

# Return the compiled model ready for training
return model
```

Figure 4 – Adjustment of build_model

Units in the final dense layer is now set to steps_ahead, which allows the model to output predictions for multiple future days in a single pass.

The addition of multivariate and multistep required the experiments to have their model configuration altered to accommodate the change. The experiments can choose between single-step and multistep predictions as well as univariate and multivariate data.

```
{
    'cell': LSTM,
    'units': 100,
    'n_layers': 3,
    'dropout': 0.1,
    'bidirectional': False,
    'epochs': 50,
    'batch_size': 16,
    'optimizer': 'adam',
    'loss': 'mean_absolute_error',
    'multistep': True,  # Multistep prediction
    'steps_ahead': STEPS_AHEAD,  # Predicting 30 days ahead
    'multivariate': True,
    'task': 'Multistep Multivariate'
}
```

Figure 5 – Adjusted experiment config

Plotting and evaluation of the multistep predictions required reshaping and scaling the predictions appropriately to predict multiple future days. For multistep predictions, the predicted values are reshaped and scaled back to their original values using the scalers, ensuring correct evaluation in comparison to the real values of the stock.

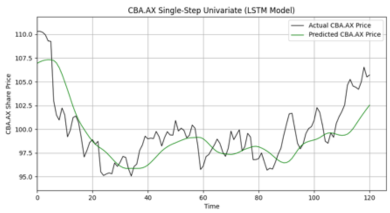
```python
        # If multistep predictions (multiple days at once), data needs to be reshaped
        if exp['multistep']:
            # Reshape the predicted prices and convert them back to their original scale.
            predicted_prices_reshaped = predicted_prices.reshape(-1, 1)
            predicted_prices_inv = scalers['Close'].inverse_transform(predicted_prices_reshaped)
            predicted_prices_inv = predicted_prices_inv.reshape(predicted_prices.shape)

            # Same for the real prices from the test set --> convert them back to the original scale.
            actual_prices_reshaped = y_test.reshape(-1, 1)
            actual_prices_inv = scalers['Close'].inverse_transform(actual_prices_reshaped)
            actual_prices_inv = actual_prices_inv.reshape(y_test.shape)
```

Figure 5 – Multistep reshaping

Data results

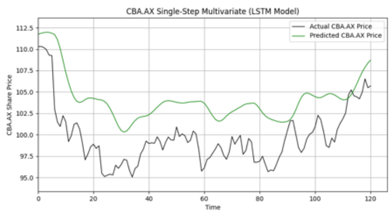
Singlestep Univariate

```python
'cell': LSTM,
'units': 50,
'n_layers': 3,
'dropout': 0.2,
'bidirectional': False,
'epochs': 25,
'batch_size': 32,
'optimizer': 'adam',
'loss': 'mean_squared_error',
'multistep': False,  # Single-step prediction
'steps_ahead': 1,  # Predicting 1 day ahead
'multivariate': False,  # Univariate (using only 'Close')
'task': 'Single-Step Univariate'
```



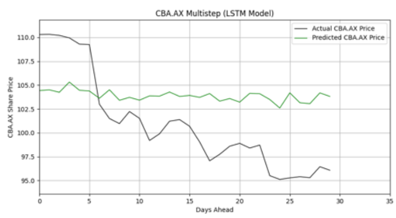
Singlestep Multivariate

```python
'cell': LSTM,
'units': 128,
'n_layers': 4,
'dropout': 0.2,
'bidirectional': False,
'epochs': 40,
'batch_size': 64,
'optimizer': 'adam',
'loss': 'mean_squared_error',
'multistep': False,
'steps_ahead': 1,
'multivariate': True,
'task': 'Single-Step Multivariate'
```



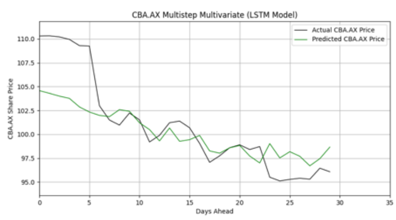
Multistep

```python
'cell': LSTM,
'units': 128,
'n_layers': 4,
'dropout': 0.2,
'bidirectional': False,
'epochs': 40,
'batch_size': 64,
'optimizer': 'adam',
'loss': 'mean_squared_error',
'multistep': True,
'steps_ahead': STEPS_AHEAD,
'multivariate': False,
'task': 'Multistep'
```



Multistep Multivariate

```python
'cell': LSTM,
'units': 128,
'n_layers': 3,
'dropout': 0.2,
'bidirectional': False,
'epochs': 50,
'batch_size': 32,
'optimizer': 'adam',
'loss': 'mean_absolute_error',
'multistep': True,  # Multistep prediction
'steps_ahead': STEPS_AHEAD,  # Predicting 30 days ahead
'multivariate': True,
'task': 'Multistep Multivariate'
```



References

Roy, B. R. (2020, October 14). Multivariate multistep LSTM. Medium. *https://bobrupakroy.medium.com/multivariate-multistep-lstm-38d9536a6b2e*

StatQuest with Josh Starmer. (2019, April 11). Introduction to recurrent neural networks: Time series prediction with LSTMs [Video]. YouTube. *https://www.youtube.com/watch?v=la2LKpf5eSU*